Media Over QUIC (MoQ) Protocol

Comprehensive Technical Guide

with Illustrated Diagrams

A Complete Reference for Research and Implementation

February 24, 2026

Contents

Executive Summary

Media over QUIC (MoQ) is a next-generation live media transport protocol being standardized by the IETF. It combines the low-latency interactivity of WebRTC with the massive scalability of HLS/DASH, all built on the modern QUIC transport protocol. MoQ is designed to be the foundational technology for real-time media delivery on the internet, offering sub-second latency at global scale.

Unlike existing protocols that require trade-offs between latency, scale, and complexity, MoQ provides a unified architecture that excels in all three dimensions. It achieves this through a publish-subscribe model built on QUIC's multiplexing capabilities, with explicit support for relay networks and CDN infrastructure.

Key Advantages

- Sub-second latency comparable to WebRTC
- Massive scalability through relay networks and CDN support
- Single protocol for both ingest and distribution
- Native browser support via WebTransport
- Intelligent congestion handling with prioritization
- No head-of-line blocking across independent streams
- Seamless connection migration during network changes

1 Introduction to MoQ

1.1 What is Media over QUIC?

Media over QUIC (MoQ) is an emerging IETF standard protocol designed specifically for real-time media streaming. At its core, MoQ treats media as subscribable tracks in a publish-subscribe system, allowing publishers to announce media tracks and subscribers to request them by name. This fundamentally different approach from existing protocols enables MoQ to achieve both interactivity and massive scale.

MoQ operates over QUIC (Quick UDP Internet Connections), the same modern transport protocol that powers HTTP/3. By building on QUIC rather than TCP, MoQ gains significant advantages including multiplexed streams without head-of-line blocking, built-in encryption, connection migration, and fast connection establishment.

1.2 The Promise of MoQ

MoQ promises to solve a long-standing problem in internet media delivery: the forced trade-off between latency and scale. Traditional solutions have required choosing between:

- **WebRTC**: Low latency (sub-second) but complex peer-to-peer architecture that doesn't scale well
- **HLS/DASH**: Massive scale through CDNs but 3-30 second latency
- **RTMP**: Simple but TCP-based with head-of-line blocking issues

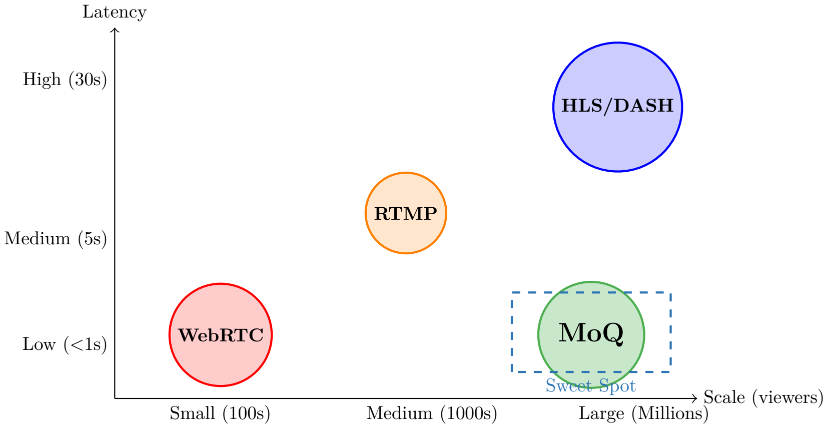


Figure 1: MoQ eliminates the latency-scale trade-off

MoQ eliminates these trade-offs by providing a single protocol that achieves sub-second latency while maintaining the scalability and economic efficiency of CDN infrastructure. This is accomplished through an intelligent relay network architecture where relays can cache and forward media without needing to decrypt it.

1.3 Protocol Family Overview

MoQ is actually a family of specifications rather than a single protocol:

- **MoQ Transport (MoQT)**: The core transport protocol that handles publish-subscribe signaling and data delivery
- **Streaming Formats**: Higher-level specifications that define how specific media types are packaged (e.g., WARP, CMAF-based formats)
- **Catalog Format**: Metadata describing available tracks and their relationships

This layered architecture allows MoQ to remain media-agnostic at the transport level while supporting various encoding formats at the application level.

2 Historical Context and Motivation

2.1 Evolution of Streaming Protocols

Understanding MoQ requires understanding the evolution of internet streaming:

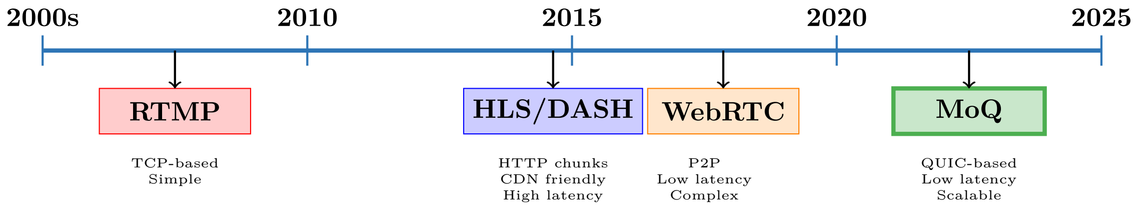


Figure 2: Evolution of streaming protocols over time

Early 2000s - RTMP Era: Real-Time Messaging Protocol (RTMP) dominated live streaming. Built on TCP, it provided simple reliable delivery but suffered from head-of-line blocking. A single lost packet would block all subsequent data until retransmitted.

2010s - HTTP Adaptive Streaming: HLS (Apple) and DASH (MPEG) revolutionized streaming by chunking media into segments served over HTTP. This enabled CDN caching and massive scale but introduced 6-30 second latency due to segment buffering.

Mid-2010s - WebRTC: Designed for real-time peer-to-peer communication with sub-second latency. However, its complexity and lack of CDN-friendly architecture made it difficult to scale beyond small groups.

Late 2010s - Low Latency HLS/DASH: Attempts to reduce HLS/DASH latency to 2-5 seconds through smaller chunks and delta manifests, but fundamental architectural limitations remained.

2.2 Why Now? The QUIC Revolution

MoQ became viable because of QUIC. Standardized as RFC 9000 in 2021, QUIC is a UDP-based transport protocol that provides:

- Stream multiplexing without head-of-line blocking
- 0-RTT connection establishment (after first connection)
- Built-in TLS 1.3 encryption
- Connection migration across network changes
- Better congestion control than TCP

HTTP/3, built on QUIC, has already been deployed widely. MoQ leverages this infrastructure through WebTransport, a W3C standard that exposes QUIC's capabilities to browser applications.

3 Core Concepts and Architecture

3.1 Core Idea

Instead of a server pushing media to each viewer (classic HLS/DASH) or a mesh of peers with NAT traversal constraints (WebRTC), MoQ allows to define **media as named publishable objects** and let many consumers **subscribe**optionally through **relays**with very low latency and high fanout.

3.2 The Three-Layer Architecture

MoQ uses a three-layer architecture that cleanly separates concerns:

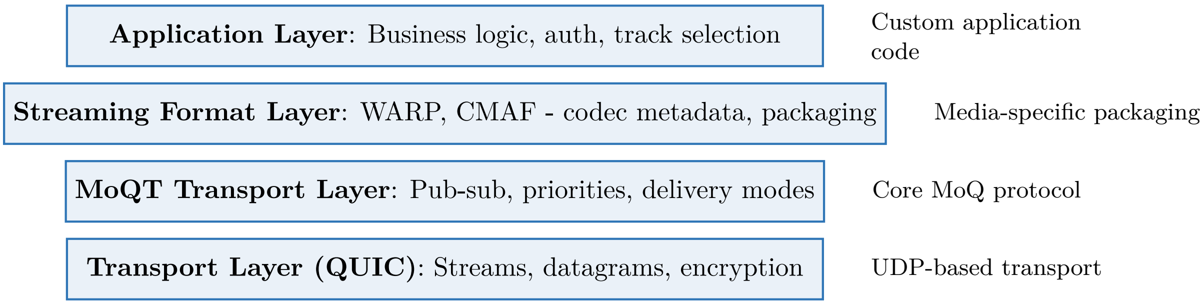
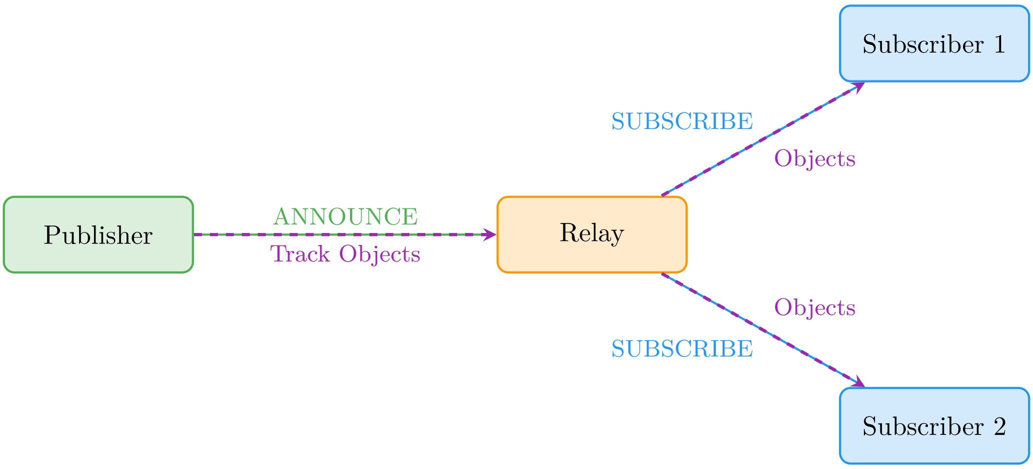

Application Layer: Business logic, auth, track selection
Custom application code
Streaming Format Layer: WARP, CMAF - codec metadata, packaging
Media-specific packaging
MoQT Transport Layer: Pub-sub, priorities, delivery modes
Core MoQ protocol
Transport Layer (QUIC): Streams, datagrams, encryption
UDP-based transport

Figure 3: MoQ Three-Layer Architecture

3.3 Publish-Subscribe Paradigm

Unlike request-response protocols (HTTP) or complex signaling protocols (WebRTC), MoQ uses a simple publish-subscribe model:



Flow: 1) Publisher ANNOUNCEs tracks
2) Subscribers SUBSCRIBE to tracks
3) Relay forwards objects to all subscribers

Figure 4: Publish-Subscribe Model

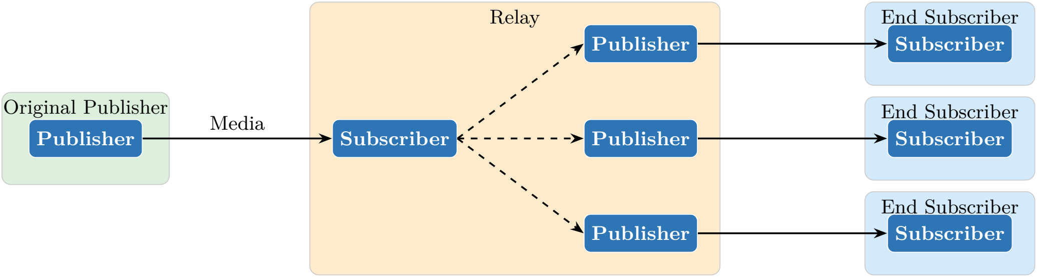
This model naturally supports one-to-many distribution and enables efficient relay forwarding without requiring complex session management.

3.4 Participants in the Correspondence

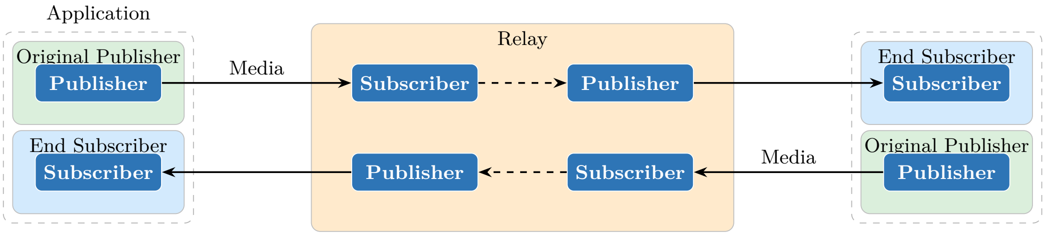
There are mainly the following three types of participants in MoQ communication.

- Original Publisher: Start sending media
- Relay: Receive media, cache it if necessary, and forward it to other participants
- End Subscriber: Receive media and do not forward it to other participants

Relay becomes a Subscriber for Original Publisher and Publisher for End Subscriber.



To send media in both directions, one can do it by replacing Original Publisher and End Subscriber for each media in the application.



3.5 Network Topology

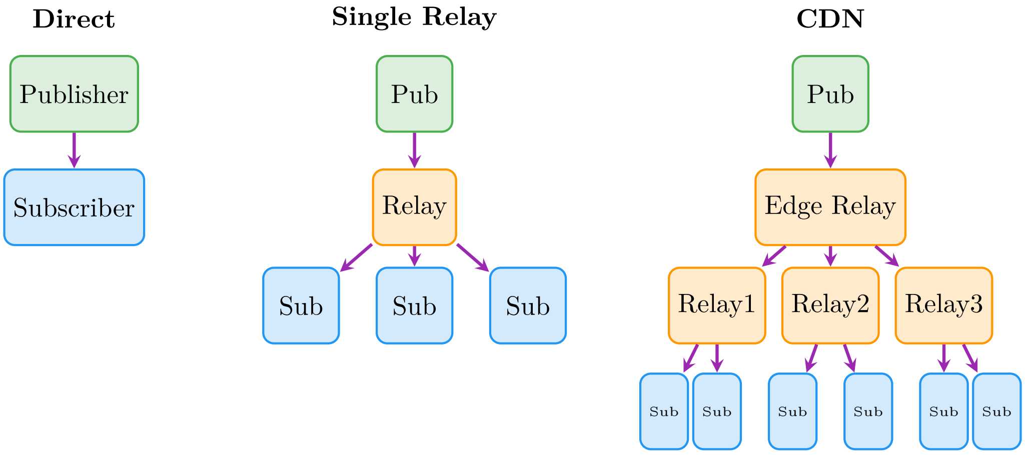


Figure 5: MoQ Network Topologies: Direct, Single Relay, and Multi-tier CDN

MoQ supports three primary deployment patterns:

Direct Connection: Publisher connects directly to subscriber. Suitable for small-scale applications or peer-to-peer scenarios.

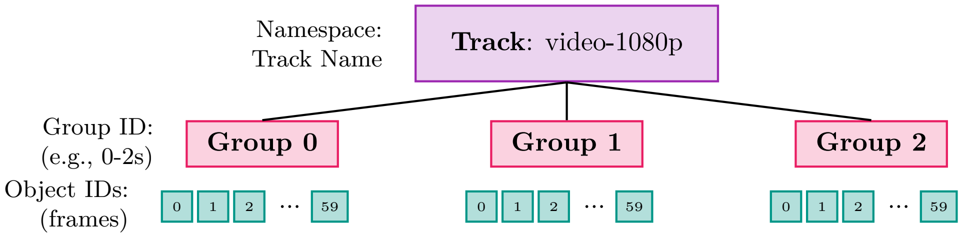
Single Relay: Publisher sends to a relay, which forwards to multiple subscribers. The relay performs fan-out, reducing publisher bandwidth.

Relay Network: Multiple interconnected relays form a tree structure. Publishers send to edge relays, which forward to other relays and eventually to subscribers. This is the standard CDN deployment model.

4 The MoQ Data Model

4.1 Hierarchical Structure

MoQ organizes media in a strict hierarchy: Tracks contain Groups, Groups contain Objects, and Objects can optionally contain Subgroups.



Hierarchy: Track ➔ Groups (GOPs) ➔ Objects (frames)
Example: Group 0 contains keyframe (object 0) + 59 P-frames
Join Points: Subscribers can join at start of any group

Figure 6: MoQ Data Model: Track, Group, and Object Hierarchy

Element	Description
Track	A named sequence of groups representing a media stream. Identified by Track Namespace + Track Name. The primary subscription target.
Group	A temporal sequence of objects that represents a join point. For video, typically corresponds to a GOP (Group of Pictures) starting with a keyframe.
Object	The basic data element - an addressable unit whose payload is a sequence of bytes. Object contents are immutable.
Subgroup	Optional subdivision within a group for priority differentiation. Used for temporal scalability.

Table 1: MoQ Data Model Elements

4.2 Track Naming

Every track in MoQ has a two-part identifier:

- **Track Namespace**: An ordered tuple of 1-32 byte sequences
- **Track Name**: A single byte sequence

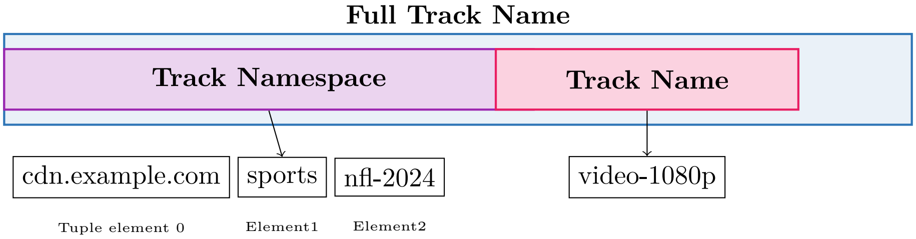


Figure 7: Track Naming Structure

4.3 Practical Example: Video Stream

Consider a live video stream with multiple quality levels:

```
Track: ['mycdn.com', 'live', 'event-123'], 'video-1080p'
  Group 0 (0-2s): Objects 0-60 (keyframe + 59 P-frames)
  Group 1 (2-4s): Objects 0-60 (keyframe + 59 P-frames)
  Group 2 (4-6s): Objects 0-60 (keyframe + 59 P-frames)

Track: ['mycdn.com', 'live', 'event-123'], 'video-720p'
  Group 0 (0-2s): Objects 0-60
  Group 1 (2-4s): Objects 0-60
```

5 QUIC Foundation

5.1 Why QUIC?

QUIC (Quick UDP Internet Connections) is a UDP-based transport protocol standardized in **RFC 9000**. MoQ builds on QUIC because it solves fundamental problems that plagued earlier streaming protocols.

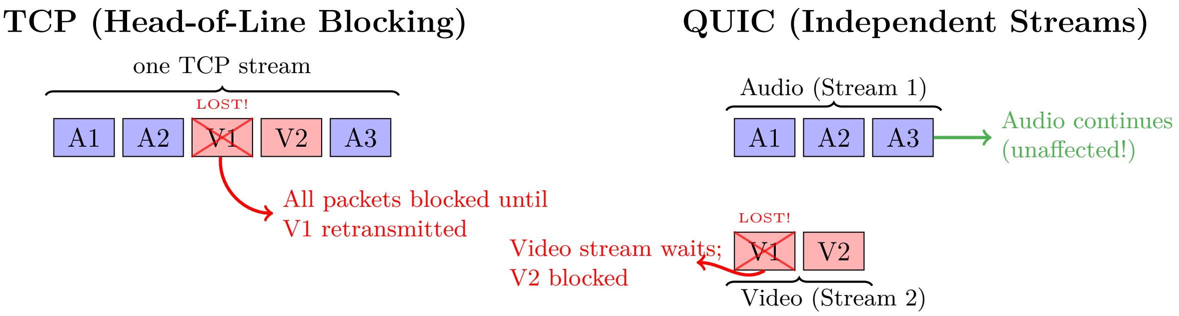


Figure 8: TCP vs QUIC: Head-of-Line Blocking

5.2 Key QUIC Features for MoQ

5.2.1 Stream Multiplexing without Head-of-Line Blocking

QUIC supports multiple independent streams within a single connection. Unlike TCP, where packet loss blocks all streams, QUIC streams are independent. If packets are lost on one stream (e.g., audio), other streams (e.g., video) continue unaffected.

5.2.2 Fast Connection Establishment

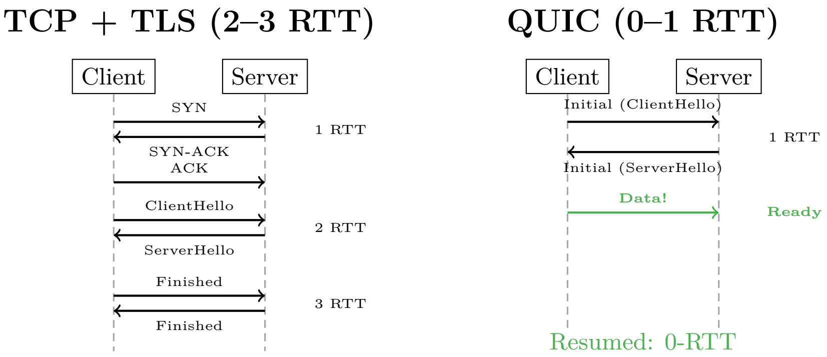
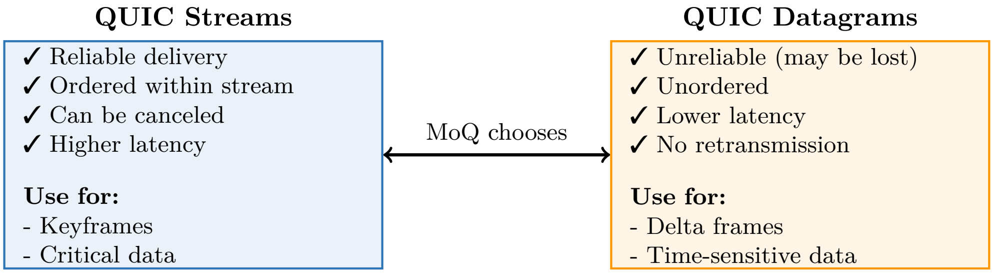


Figure 9: Connection Establishment: TCP+TLS vs QUIC

5.3 Streams vs. Datagrams

QUIC provides two transport mechanisms that MoQ can use:



QUIC Streams

✓ Reliable delivery
✓ Ordered within stream
✓ Can be canceled
✓ Higher latency

Use for:
- Keyframes
- Critical data

MoQ chooses

QUIC Datagrams

✓ Unreliable (may be lost)
✓ Unordered
✓ Lower latency
✓ No retransmission

Use for:
- Delta frames
- Time-sensitive data

Figure 10: QUIC Streams vs Datagrams

5.4 WebTransport: QUIC in Browsers

WebTransport is a W3C specification that exposes QUIC's capabilities to browser JavaScript. It provides APIs for:

- Opening bidirectional and unidirectional streams
- Sending datagrams
- Accessing connection state and errors

MoQ implementations can use either native QUIC (for servers and native apps) or WebTransport (for browser clients). The protocol works identically over both transports, with WebTransport adding an HTTP/3 handshake layer for browser compatibility.

Browser Support (as of early 2025):

- Chrome/Edge: Full WebTransport support
- Firefox: Experimental support (flag required)
- Safari: Under development, not yet available

6 MoQ Transport (MoQT) Protocol

6.1 Connection Establishment

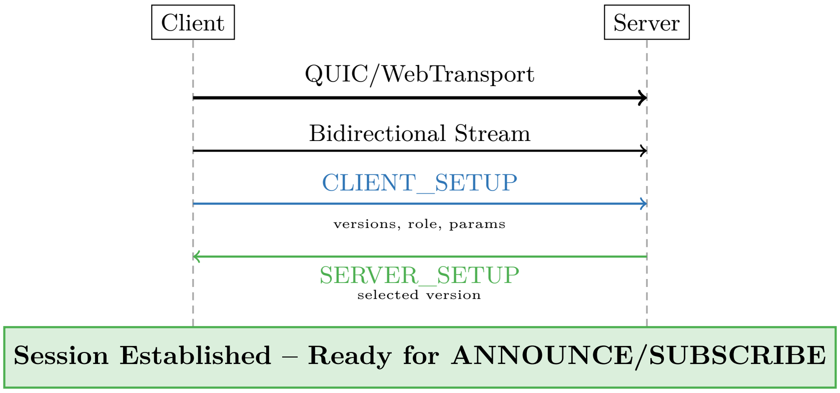


Figure 11: MoQT Session Establishment

6.2 Control Messages

Message	Purpose
`CLIENT_SETUP`	Initiate session, propose versions and parameters
`SERVER_SETUP`	Accept session, select version
`ANNOUNCE`	Publisher announces availability of track namespace
`ANNOUNCE_OK`	Relay accepts announcement
`SUBSCRIBE`	Subscriber requests a track with filter parameters
`SUBSCRIBE_OK`	Subscription accepted, data will flow
`UNSUBSCRIBE`	Cancel existing subscription
`FETCH`	Request specific historical groups (for catch-up)
`GOAWAY`	Graceful session termination

Table 2: MoQT Control Messages

6.3 Object Delivery Modes

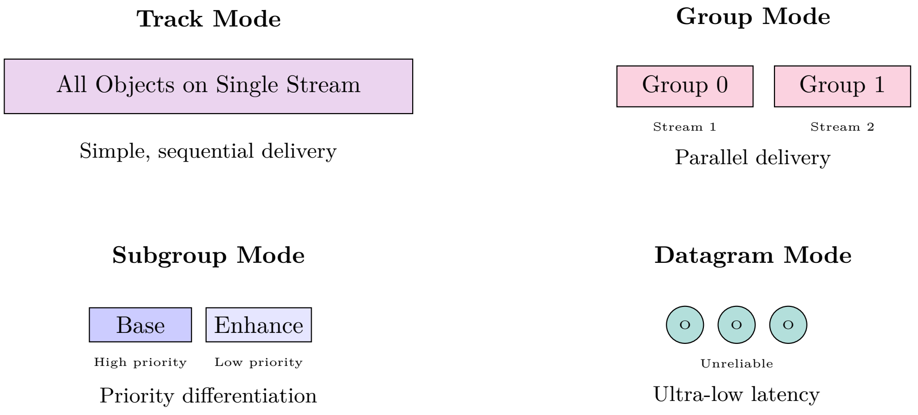
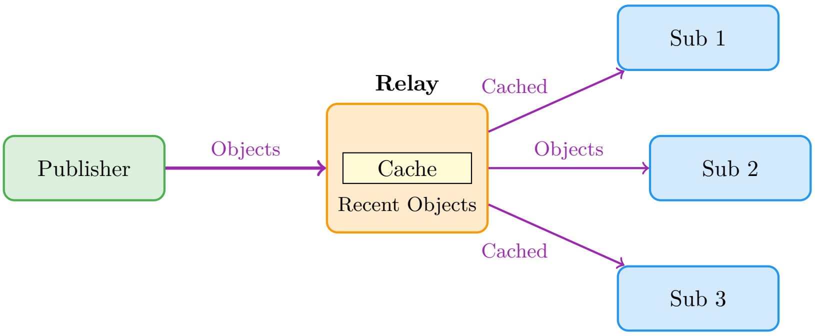


Figure 12: MoQT Object Delivery Modes

7 Relay Networks and CDN Architecture

7.1 Relay Role and Caching

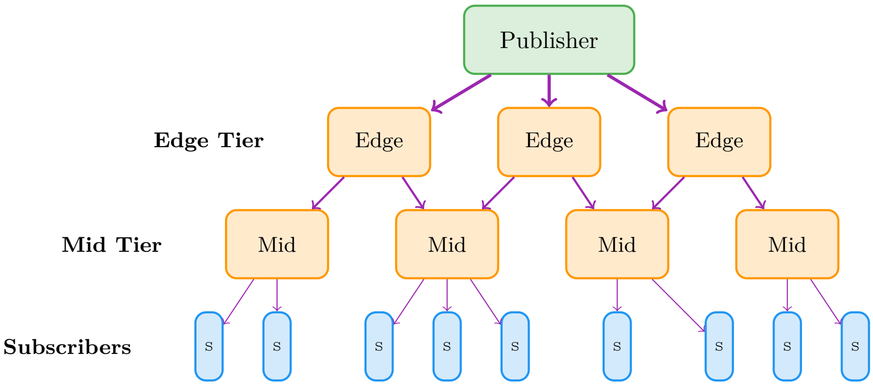


Relay Functions:

- ● Receives objects from upstream
- ● Caches recent objects
- ● Forwards to multiple subscribers
- ● Enables efficient fan-out

Figure 13: Relay Caching and Fan-out

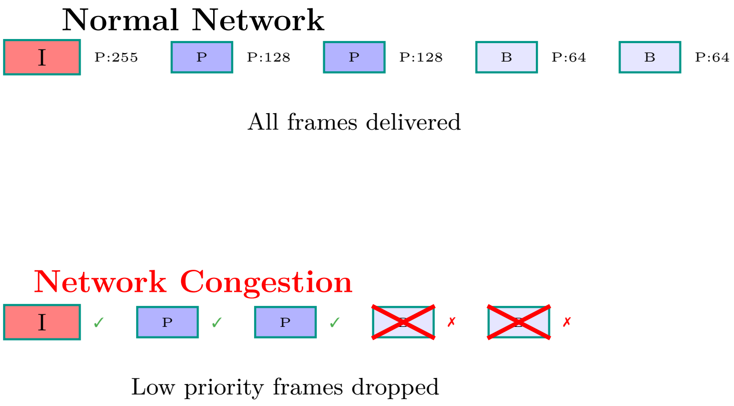
7.2 Multi-tier CDN Architecture



Geographic distribution: Each tier serves nearby regions for minimal latency

Figure 14: Multi-tier CDN Relay Network

8 Priority and Congestion Management



Priority Values: I-frame=255 (highest), P-frame=128, B-frame=64 (lowest)
Graceful Degradation: Lower quality maintained vs complete stall

Figure 15: Priority-based Congestion Response

9 Comparison with Existing Protocols

Feature	MoQ	WebRTC	HLS/DASH	RTMP
Latency	<1 second	<1 second	3-30 seconds	3-5 seconds
Scale	Millions	Hundreds	Millions	Thousands
Transport	QUIC	UDP/SRTP	HTTP/TCP	TCP
CDN Support	Native	Complex	Native	Moderate
Browser Support	WebTransport	Native	Native	Flash only
Complexity	Moderate	Very High	Low	Low
HOL Blocking	No	No	Yes	Yes
Connection Migration	Yes	No	No	No

Table 3: Detailed Protocol Comparison

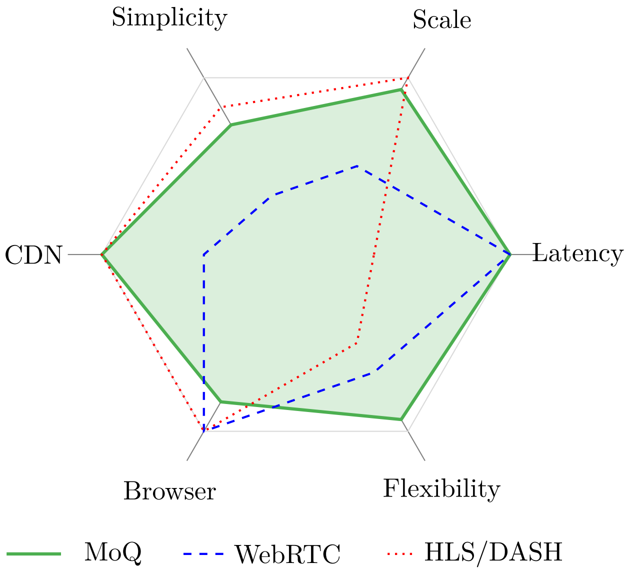


Figure 16: Protocol Capability Comparison (higher = better)

10 Implementation Guide

10.1 Available Libraries

Several open-source implementations exist:

- **moq-rs (Rust)**: Reference implementation by kixelated
- **moq-js (TypeScript)**: Browser-compatible library
- **moqtransport (Go)**: By mengelbart
- **MediaMTX (Go)**: Streaming server with MoQ support

10.2 Implementation Workflow

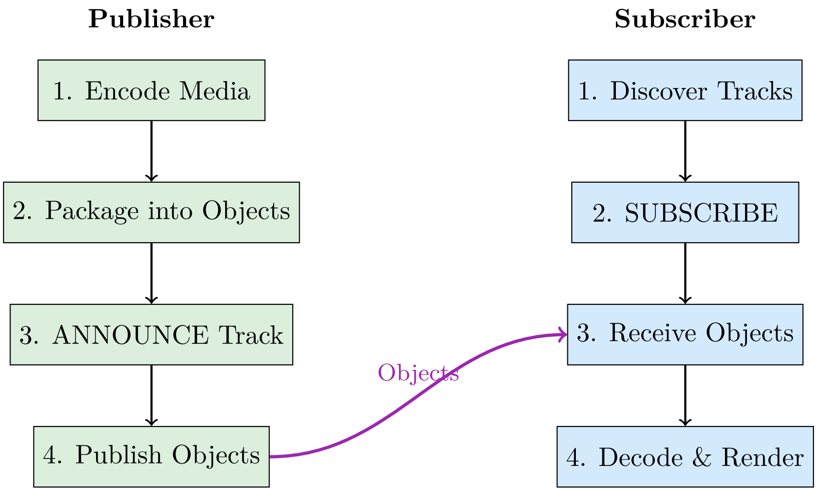


Figure 17: Publisher and Subscriber Implementation Workflow

11 Use Cases and Applications

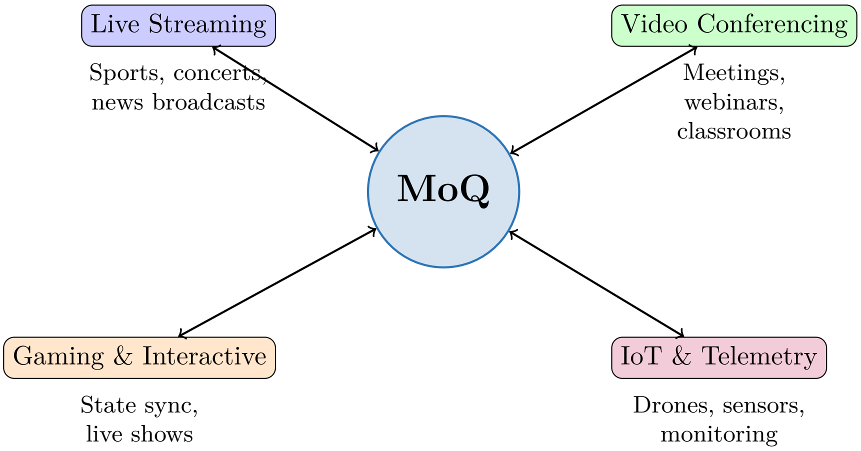


Figure 18: MoQ Use Cases

Conclusion

Media over QUIC represents a fundamental shift in how we think about real-time media delivery on the internet. By building on QUIC's solid transport foundation and introducing an elegant publish-subscribe model, MoQ eliminates the traditional trade-offs between latency, scale, and complexity.

The protocol's design reflects lessons learned from decades of streaming protocol evolution. From RTMP's simplicity but TCP limitations, through HLS/DASH's massive scale but high latency, to WebRTC's low latency but complexity, MoQ synthesizes the best aspects of each while avoiding their pitfalls.

For researchers, MoQ offers rich opportunities to explore optimization strategies, develop new streaming formats, and investigate the interaction between application-level priorities and transport-level congestion control. The protocol's clean layering and extensible design make it an excellent platform for experimentation.

As the standardization process continues and implementations mature, MoQ is positioned to become the foundation for next-generation streaming applications. Whether you're building a live streaming platform, video conferencing system, or real-time collaborative tool, understanding MoQ will be essential.

Recommended Resources

To continue your MoQ journey:

- IETF MoQ Working Group: https://datatracker.ietf.org/wg/moq/
- MoQ Transport Draft: https://datatracker.ietf.org/doc/draft-ietf-moq-transport/
- moq-rs Repository: https://github.com/kixelated/moq-rs
- Cloudflare MoQ Blog: https://blog.cloudflare.com/moq/
- MoQ Developer Site: https://moq.dev/
- Ali C. Begen - Streaming Bad: Breaking Latency with MOQ: https://www.youtube.com/watch?v=l4dwczI5Bis

The protocol continues to evolve rapidly. Stay engaged with the community, experiment with implementations, and contribute to the ecosystem. The future of real-time media on the internet is being built right now.